UNITED STATES PATENT APPLICATION

FOR

Method and apparatus for Control of Rate-Distortion Tradeoff by Mode Selection in Video Encoders

Inventors: Barin G. Haskell
Adriana Dumitras
Atul Puri

Prepared by:
Dag Johansen of Stattler, Johansen & Adeli LLP
P.O. Box 51860
Palo Alto, California 94303-0728
PHONE: 650.934.0470 X101
FAX: 650.934.0475

Method and Apparatus for Control of Rate-Distortion Tradeoff by Mode Selection in Video Encoders

RELATED APPLICATIONS

The present patent application claims the benefit of the previous U.S. Provisional Patent Application entitled "**Method and apparatus for Control of Rate-Distortion Tradeoff by Mode Selection in Video Encoders**" having serial number 60/424,738 that was filed on November 7, 2002.

FIELD OF THE INVENTION

The present invention relates to the field of multi-media compression and encoding systems. In particular the present invention discloses methods and systems for controlling the rate-distortion tradeoff in a digital video encoder.

BACKGROUND OF THE INVENTION

Digital based electronic media formats are completely replacing the legacy analog electronic media formats. In the audio domain, digital compact discs (CDs) replaced analog vinyl recordings many years ago. Analog magnetic cassette tapes are becoming increasingly rare. Second and third generation digital audio systems such as Mini-discs and MP3 (MPEG Audio - layer 3) based formats are now taking away market share from the first generation digital audio format of compact discs.

Film-based still photography is rapidly being replaced by digital still photography. Immediate image availability and image distribution via the Internet have provided users with irresistible features.

However, the video domain has been slower to move to digital storage and transmission formats than audio and still images. This has been largely due to the massive amounts of digital information required to accurately represent video in digital form. The massive amounts of digital information needed to accurately represent video require very high-capacity digital storage systems and high-bandwidth transmission systems.

But the video domain is finally adopting digital storage and transmission formats. Faster computer processors, high-density storage systems, high-bandwidth optical transmission lines, and new efficient video encoding algorithms have finally made digital video systems practical at consumer price points. The DVD (Digital Versatile Disc), a digital video system, has been one of the fastest selling consumer electronic products ever. DVDs have been rapidly supplanting Video-Cassette Recorders (VCRs) as the pre-recorded video playback system of choice due their exceptional video quality, high quality 5.1 channel digital audio, convenience, and extra features. In the realm of video transmission systems, the antiquated analog NTSC (National Television Standards Committee) video transmission standard is finally being replaced with the digital ATSC (Advanced Television Standards Committee) video transmission system that uses digital compression and encoding technology.

Computer systems have been using various different digital video encoding formats for a number of years. Among the best digital video compression and encoding systems used by computer systems have been the digital video systems backed by the Motion Pictures Expert Group commonly known by the acronym MPEG. The three most well known and highly used digital video formats from MPEG are known simply as MPEG-1, MPEG-2, and MPEG-4. Video CDs and consumer-grade digital video editing systems use the early MPEG-1 format. Digital Versatile Discs (DVDs) and the Dish Network brand Direct Broadcast Satellite (DBS) television broadcast system use the MPEG-2 digital video compression and encoding system. The MPEG-4 encoding system is rapidly being adapted by the latest computer based digital video encoders and associated digital video players.

The MPEG-2 and MPEG-4 standards compress a series of video frames or video fields and then encode the compressed frames or fields into a digital bitstream. The rate of the digital bitstream must be carefully monitored in order not to overflow memory buffers, underflow memory buffers, or exceed the transmission channel capacity. Thus, a sophisticated rate control system must be implemented with the digital video encoder that provides the best possible image quality in the allotted channel capacity without overflowing or underflowing buffers.

SUMMARY OF THE INVENTION

A Method And Apparatus For Control of Rate-Distortion Tradeoff by Mode Selection in Video Encoders is Disclosed. The system of the present invention first selects a distortion value D near a desired distortion value. Next, the system determines a quantizer value Q using the selected distortion value D. The system then calculates a Lagrange multiplier lambda using the quantizer value Q. Using the selected Lagrange multiplier lambda and quantizer value Q, the system begins encoding pixelblocks.

If the system detects a potential buffer overflow, then the system will increase the Lagrange multiplier lambda. The potential buffer overflow may be detected when a memory buffer occupancy value exceeds an overflow threshold value. If the Lagrange multiplier lambda exceeds a maximum lambda threshold then the system will increase the quantizer value Q.

If the system detects a potential buffer underflow, then the system will decrease the Lagrange multiplier lambda. The potential buffer underflow may be detected when a memory buffer occupancy value falls below a buffer underflow threshold value. If the Lagrange multiplier lambda falls below a minimum lambda threshold then the system will decrease the quantizer value Q.

Other objects, features, and advantages of present invention will be apparent from the company drawings and from the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features, and advantages of the present invention will be apparent to one skilled in the art, in view of the following detailed description in which:

Figure 1 illustrates a high-level block diagram of one possible a digital video encoder system.

Figure 2 illustrates a series of video pictures in the order that the pictures should be displayed wherein the arrows connecting different pictures indicate inter-picture dependency created using motion compensation.

Figure 3 illustrates the series of video pictures from **Figure 2** rearranged into a preferred transmission order of video pictures wherein the arrows connecting different pictures indicate inter-picture dependency created using motion compensation.

Figure 4 graphically illustrates a family of R,D curves, with one curve for each different value of quantizer Q .

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Modalities to control the rate-distortion tradeoff by mode selection in video encoders are disclosed. In the following description, for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. For example, the present invention has been described with reference to the MPEG-4 Part 10 (H.264) multimedia compression and encoding system. However, the same techniques can easily be applied to other types of compression and encoding systems.

Multimedia Compression and Encoding Overview

Figure 1 illustrates a high-level block diagram of a typical digital video encoder **100** as is well known in the art. The digital video encoder **100** receives incoming video stream **105** at the left of the block diagram. Each video frame is processed by a Discrete Cosine Transformation (DCT) unit **110**. The video frame may be processed independently (an intra-frame) or with reference to information from other frames (an inter-frame) using motion estimation unit **160**. A Quantizer (Q) unit **120** then quantizes the information from the Discrete Cosine Transformation unit **110**. The quantized frame is then encoded with an entropy encoder (H) unit **180** to produce an encoded video bitstream.

Since an inter-frame encoded video frame is defined with reference to other nearby video frames, the digital video encoder **100** needs to create a copy of exactly

how a referenced digital video frame will appear within a digital video decoder such that inter-frames may be encoded. Thus, the lower portion of the digital video encoder **100** is actually a digital video decoder. Specifically, Inverse quantizer (Q^{-1}) **130** reverses the quantization of the frame information and inverse Discrete Cosine Transformation (DCT^{-1}) unit **140** reverses the Discrete Cosine Transformation of the video frame information. After all the DCT coefficients are reconstructed from inverse Discrete Cosine Transformation, the motion compensation unit will use the information, along with the motion vectors, to reconstruct the video frame which then may be used as a reference video frame for the motion estimation of other video frames.

The decoded video frame may be used to encode inter-frames that are defined relative to information in the decoded video frame. Specifically, a motion compensation (MC) unit **150** and a motion estimation (ME) unit **160** are used to determine motion vectors and generate differential values used to encode inter-frames.

A rate controller **190** receives information from many different components in a digital video encoder **100** and uses the information to allocate a bit budget for each video frame to be encoded. The bit budget should be assigned in a manner that will generate the highest quality digital video bit stream that complies with a specified set of restrictions. Specifically, the rate controller **190** attempts to generate the highest quality of a compressed video stream without overflowing memory buffers (exceeding the amount of available buffer memory by sending video frame information much faster than the video frame information is displayed and subsequently deleted) or underflowing memory buffers (not sending video frame information fast enough such that a receiving digital video decoder runs out of video frame information to display).

Pixelblock Encoding

Many digital video coding algorithms first partition each video picture into small subsets of pixels that are generally referred to as pixelblocks. Specifically, the video picture is divided into a grid of rectangular pixelblocks. The terms Macroblock, block, sub-block are also commonly used for subsets of pixels. This document will use the term pixelblock to include all of these different but similar constructs. Different sized pixelblocks may be used by different digital video encoding systems. For example, different pixelblock sizes used include 8 pixel by 8 pixel pixelblocks, 8 pixel by 4 pixel pixelblocks, 16 pixel by 16 pixel pixelblocks, 4 pixel by 4 pixel pixelblocks, etc.

To encode a video picture, each individual pixelblock of the video picture is encoded using some sort of encoding method. Some pixelblocks known as intra-blocks are encoded without reference to any other pixelblocks. Other pixelblocks are encoded using some predictive coding method such as motion compensation that refers to a closely matching pixelblock in the same or a different video picture.

Each individual pixelblock in a video picture is independently compressed and encoded. Some video coding standards, e.g., ISO MPEG or ITU H.264, use different types of predicted pixelblocks to encode digital video pictures. In one scenario, a pixelblock may be one of three types:

1. I-pixelblock – An Intra (I) pixelblock uses no information from any other video pictures in its coding (Thus, an intra-pixelblock it is completely self-defined.);

2. P-pixelblock – A unidirectionally Predicted (P) pixelblock refers to picture information from an earlier video picture; or
3. B-pixelblock – A Bi-directional predicted (B) pixelblock uses information from both an earlier video picture and a later future video picture.

If all of the pixelblocks in an encoded digital video picture are Intra-pixelblocks (I-pixelblocks) then the encoded digital video picture frame is known as an Intra-frame. Note that an Intra-frame makes no reference to any other video picture such that the Intra-frame digital video picture is completely self-defined.

If a digital video picture frame only includes unidirectional predicted pixelblocks (P-pixelblocks) and intra-pixelblocks (I-pixelblocks) but no bi-directional predicted pixelblocks (B-pixelblocks), then the video picture frame is known as a P-frame. I-pixelblocks may appear in P-frames when using predictive encoding (P-Pixelblock encoding) requires more bits than an independently encoded pixelblock (an I-pixelblock).

If a digital video picture frame contains any bi-directional predicted pixelblocks (B-pixelblocks), then the video picture frame is known as a B-frame. For the simplicity, this document will consider the case where all pixelblocks within a given picture are of the same type. (Intra-frames only contain I-pixelblocks, P-frames only contain P-pixelblocks, and B-frames only contain B-pixelblocks.)

An example sequence of video pictures to be encoded might be represented as:

$I_1\ B_2\ B_3\ B_4\ P_5\ B_6\ B_7\ B_8\ B_9\ P_{10}\ B_{11}\ P_{12}\ B_{13}\ I_{14}\ldots$

where the letter (I, P, or B) represents if the digital video picture frame is an I-frame, P-frame, or B-frame and the numerical subscript represents the camera order of the video picture in the sequence of video pictures. The camera order is the order in which a camera recorded the video pictures and thus is also the order in which the video pictures should be displayed (the display order).

The preceding example series of video pictures is conceptually illustrated in **Figure 2**. Referring to **Figure 2**, the arrows indicate that pixelblocks from a stored picture (I-frame or P-frame in this case) are used in the motion compensated prediction of other digital video pictures (B-frames and P-frames).

Referring to **Figure 2**, no information from any other video picture is used in the encoding of the first video picture frame, intra-frame video picture I_1. Video picture P_5 is a P-frame that uses video information from previous video picture I_1 in its encoding such that an arrow is drawn from intra-frame video picture I_1 to P-frame video picture P_5. Video picture B_2, video picture B_3, and video picture B_4 all use information from both video picture I_1 and video picture P_5 in their encoding such that information dependency arrows are drawn from video picture I_1 and video picture P_5 to video picture B_2, video picture B_3, and video picture B_4.

Since B-frame video pictures use information from later video pictures (pictures that will be displayed later), the transmission order of a set of digital video

pictures is usually different than the display order of the digital video pictures. Specifically, referenced video pictures that are needed to construct other video pictures should be transmitted before the video pictures that are dependent upon referenced video pictures. Thus, for the display order of **Figure 2**, the preferred transmission order might be:

$$I_1\ P_5\ B_2\ B_3\ B_4\ P_{10}\ B_6\ B_7\ B_8\ B_9\ P_{12}\ B_{11}\ I_{14}\ B_{13}\ \ldots$$

Figure 3 graphically illustrates this preferred transmission order of the video pictures from **Figure 2**. Again, the arrows in the figure indicate that pixelblocks from a referenced video picture (an I-frame or P-frame video picture in this case) are used in the motion compensated prediction of other video pictures (P-frame and B-frame video pictures).

Referring to **Figure 3**, the transmitting system first transmits I-frame I_1 which does not depend on any other video frame. Next, the system transmits P-frame video picture P_5 that depends only upon previously transmitted video picture I_1. Next, the system transmits B-frame video picture B_2 after video picture P_5 even though video picture B_2 will be displayed before video picture P_5. The reason for this is that when it comes time to decode and render dependent video picture B_2, the digital video decoder will have already received and decoded the information in video picture I_1 and video picture P_5 necessary to decode dependent video picture B_2. Similarly, decoded video picture I_1 and decoded video picture P_5 are ready to be used to decode and render the next two dependent video pictures: dependent video picture B_3 and dependent video picture B_4.

The receiver/decoder system then reorders the video picture sequence for proper display. In this operation, referenced video picture I_1 and referenced video picture P_5 are referred to as "stored pictures." Stored pictures are used to reconstruct other dependent video pictures that refer to the stored pictures. (Note that some digital video encoding systems also allow B-frames to be used as stored pictures.)

P-Pictures

The encoding of P-Pictures typically utilizes Motion Compensation (MC), wherein a Motion Vector (MV) pointing to a location in a previous video picture is computed for each pixelblock in the current video picture. The Motion Vector refers to a closely matching pixelblock in a referenced video picture. Using the motion vector, a prediction pixelblock can be formed by translation of referenced pixels in the aforementioned previous video picture. The difference between the actual pixelblock in the P-Picture and the prediction pixelblock is then coded for transmission. This difference is then used to accurately reconstruct the original pixelblock.

Each motion vector may also be transmitted via a predictive encoding method. For example, a motion vector prediction may be formed using nearby motion vectors. In such a case, then the difference between the actual motion vector and a predicted motion vector is then coded for transmission. The difference is then used to create the actual motion vector from the predicted motion vector.

B-Pictures

Each B-pixelblock in a B-frame uses two different motion vectors: a first motion vector that references a pixelblock in an earlier video picture and a second motion

vector that references another pixelblock in a later video picture. From these two motion vectors, two prediction pixelblocks are computed. The two predicted pixelblocks are then combined together, using some function, to form a final predicted pixelblock. (The two predicted pixelblocks may simply be averaged together.) As with P-pixelblocks, the difference between the actual desired pixelblock for the B-frame picture and the final predicted pixelblock is then encoded for transmission. The pixelblock difference will then used to accurately reconstruct the original desired pixelblock.

As with P-pixelblocks, each motion vector (MV) of a B-pixelblock may also be transmitted via a predictive encoding method. Specifically, a predicted motion vector may be formed using some combination of nearby motion vectors. Then, the difference between the actual motion vector and the predicted is encoded for transmission. The difference is then used to recreate the actual motion vector from the predicted motion vector.

However, with B-pixelblocks the opportunity exists for interpolating motion vectors from motion vectors in the collocated or nearby stored picture pixelblock. Such motion vector interpolation is carried out both in the digital video encoder and the digital video decoder. (Remember that a digital video encoder always includes a digital video decoder.)

In some cases, the interpolated motion vector is good enough to be used without any type of correction to the interpolated motion vector. In such cases, no motion vector data need be sent. This is referred to as 'Direct Mode' in the ITU H.263 and H.264 digital video encoding standards.

The technique of motion vector interpolation works particularly well on a series of digital video pictures from a video sequence created by a camera that is slowly panning across a stationary background. In fact, such motion vector interpolation may be good enough to be used alone. Specifically, this means that no differential motion vector information needs be calculated or transmitted for these B-pixelblock motion vectors encoded using such motion vector interpolation.

Pixelblock Encoding

Within each video picture the pixelblocks may also be coded in different manners. For example, a pixelblock may be divided into smaller subblocks, with motion vectors computed and transmitted for each subblock. The shape of the subblocks may also vary and may not necessarily be square.

Within a P-picture or B-picture, some pixelblocks may be more efficiently encoded without using motion compensation if no closely matching pixelblock can be found in the stored reference picture. Such pixelblocks would then be coded as Intra-pixelblocks (I-pixelblocks). Within a B-picture, some pixelblocks may be better coded using unidirectional motion compensation instead of bi-directional motion compensation. Thus, those pixelblocks would be coded as forward predicted pixelblocks (P-pixelblocks) or backward predicted pixelblocks depending on whether the closest matching pixelblock was found in an earlier video picture or a later video picture.

Prior to transmission, the prediction error of a pixelblock or subblock is typically transformed by an orthogonal transform such as the Discrete Cosine Transform

or an approximation thereto. The result of the transform operation is a set of transform coefficients equal in number to the number of pixels in the pixelblock or subblock being transformed. At the receiver/decoder, the received transform coefficients are inverse transformed to recover the prediction error values to be used further in the decoding. Not all the transform coefficients need be transmitted for acceptable video quality. Depending on the transmission bit-rate available, more than half or sometimes much more than half of the transform coefficients may be deleted and not transmitted. At the decoder the deleted coefficient values are replaced by zero values prior to inverse transform operation.

Furthermore, prior to transmission the transform coefficients are typically Quantized and Entropy Coded as set forth with reference to **Figure 1**. Quantization involves representation of the transform coefficient values by a finite subset of possible values, which reduces the accuracy of transmission. Furthermore, the quantization often forces small transform coefficient values to zero, thus further reducing the number of transform coefficients values that are transmitted.

In the quantization step, each transform coefficient value is typically divided by a quantizer step size Q and rounded to the nearest integer. For example, the original transform coefficient C may be quantized into the quantized coefficient value Cq using the formula:

$$Cq = (C + Q/2)/Q \qquad \text{truncated to an integer.}$$

After the quantization step, the integers are then entropy coded using variable length codes (VLC) such as Huffman codes or Arithmetic codes. Since many transform

coefficient values will be truncated to zero, a good amount of compression will be achieved from the quantization and variable length coding steps.

Using a Lagrange Function to Select Bit Rate and Distortion Values

A digital video encoder must determine the best encoding method amongst all of the possible encoding methods (or encoding modes) that will be used to encode each pixelblock in a video picture. This encoding problem is commonly known as the mode selection problem. Many ad hoc solutions have been used in various digital video encoder implementations to address the mode selection problem. The combination of the transform coefficient deletion, the quantization of the transform coefficients that are transmitted, and the mode selection leads to a reduction of the bit rate R used for transmission. However, these bit rate R reduction techniques also lead to a distortion D in the decoded video pictures.

Ideally, when designing a video encoder one would like to either fix the bit rate R to a constant value and minimize the coding distortion D or fix the coding distortion D to a constant value while minimizing the bit rate R. However, especially at the pixelblock level, the bit rate R and/or the distortion D value may vary considerably from the desired fixed value, thus making the constrained optimization approach untenable.

Instead what may be done is to use a Lagrange multiplier to convert the constrained optimization problem into an unconstrained optimization problem. Thus,

instead of fixing one of the variables (bit rate R or the distortion D) and optimizing the other variable, one may instead simply minimize the Lagrange function:

$$D + \text{lambda} \times R$$

where lambda is the Lagrange multiplier. Thus, for each pixelblock in a video picture, the encoder selects the pixelblock encoding mode that minimizes the Lagrange function $D + \text{lambda} \times R$.

In theory, a full optimization for each individual video picture would be carried out by repeatedly using all possible values of lambda, with each lambda producing a {D,R} pair. From this, for a desired bit rate R (or distortion D), the corresponding distortion D (or bit rate R) and lambda value can be found. Then the video picture would be finally encoded again using this selected lambda value, which would produce the desired result.

In practice, this ideal approach is usually too complicated and too resource intensive to perform for every video picture. What is usually done is to carry out many preliminary experiments with many video pictures using the complete optimization method above with a wide range of lambdas in order to determine approximate relationships between lambda, distortion D, and quantizer Q.

Preliminary experiments with a large number of video pictures using the complete optimization method with a wide range of lambdas determine approximate relationships between lambda, distortion D, and quantizer Q. In these experiments it is often advantageous to hold the quantizer Q constant while varying the lambda Lagrange multiplier. If quantizer Q is held constant during each experiment, the end result is a

family of R,D curves, with one curve for each different value of quantizer Q. **Figure 4** illustrates one example of such a family of R,D curves. For each different constant Q curve, at a particular {R,D} point obtained with a certain value of lambda the slope of the curve is (–lambda). The optimum {R,D} relationship is obtained by taking the minimum of all the R,D curves.

Following this, for each different quantizer Q value, a representative lambda value is chosen such as $lambda_Q$. For example, $lambda_Q$ might be the value that provides a distortion D value half-way between the crossover points for Q+1 and Q-1 in **Figure 4**. Other methods that have been used to select a representative lambda value include $lambda_Q = 0.85Q^2$ and $lambda_Q = 0.85 \times 2^{Q/3}$. For multiple B-pictures, much larger $lambda_Q$ values are often chosen. Thus, we have

$lambda_Q = f(Q)$
$D_Q = g(Q)$ from which one can obtain $Q = h(D_Q)$

Then to encode a video picture sequence with a desired distortion D, one may first find the nearest D_Q from which one obtains $Q = h(D_Q)$. Then the video picture encoding is performed using the corresponding $lambda_Q = f(Q)$, which provides the optimum bit rate R for the distortion D_Q.

In many applications, the resulting bit rate R may be too large or too small necessitating the use of rate control to ensure that no buffer overflow or buffer underflow occurs. With most rate control algorithms, the usual method is to vary the quantizer Q from pixelblock to pixelblock and/or from video picture to video picture. When the encoder buffer threatens to get too full (and possibly overflow) the value of the quantizer Q is increased in order to reduce the bit rate R. When the encoder buffer is too empty

(and will possibly underflow), the quantizer Q is reduced in order to increase the bit rate R.

However, the changing of the quantizer Q value may result in too large of a change in the bit rate R. Furthermore, changes in the Quantizer Q need to be signaled to the decoder, which adds to the amount of overhead bits that must be transmitted to the decoder. Moreover, changing Quantizer Q may have other effects relating to video picture quality, such as loop filtering.

An alternative to changing the quantizer Q is to change the Lagrange multiplier lambda in order to achieve the desired rate control. A smaller value of the Lagrange multiplier lambda results in a larger bit rate R (and smaller distortion D), and similarly a larger value of the Lagrange multiplier lambda decreases the bit rate R (and increases distortion D). The variation in the Lagrange multiplier lambda can be arbitrarily fine, as opposed to changes in the quantizer Q that is digitized and encoded such that the quantizer Q is limited to only certain values. In many digital video compression and encoding systems, including all of the MPEG video compression and encoding standards, not all integer values of the quantizer Q are allowed to be sent, in which case the abrupt change in bit rate R may be even more pronounced.

When the Lagrange multiplier lambda is required to be larger than a certain threshold Lambda_max(Q) to achieve a certain bit rate reduction, then the quantizer Q would be increased and the Lagrange multiplier lambda would return to its nominal value f(Q) using the newly increased quantizer Q value. When the Lagrange multiplier lambda is required to be smaller than a certain threshold Lambda_min(Q) to

achieve a certain bit rate increase, then the quantizer Q would be decreased and the Lagrange multiplier lambda would return to its nominal value f(Q) using the newly decreased quantizer Q.

The values of Lambda_max(Q) and Lambda_min(Q) are determined by the crossover points on the bit rate-distortion relationship illustrated in **Figure 4**. If one defines D(lambda,Q) to be the distortion achieved when encoding with the Lagrange multiplier lambda and quantizer step size Q, then the operative relationships are

$$D(\text{Lambda\_min}(Q+1), Q+1) = D(\text{Lambda\_max}(Q), Q)$$
$$\text{Lambda\_min}(Q) <= f(Q) <= \text{Lambda\_max}(Q)$$

The detailed operation of such a rate control algorithm for a video encoding system is set forth in the following pseudo code:

```
Start_encoding_picture:          // Begin encoding a video picture
    input desired D;             // Get the desired Distortion D value
    find D_Q nearest to D;       // Find the D_Q value closest to the desired D
    Q = h(D_Q);                  // Determine the quantizer value Q
    lambda = f(Q);               // Determine the Lagrange multiplier lambda
start_encoding_pixelblock:       // Begin encoding a pixelblock from the picture
    code_pixelblock(lambda,Q);       //Encode pixelblock using lambda and Q
    if (encoder_buffer > Tfull){     // Buffer threatens to overflow?
        lambda = lambda+deltalambda; // deltalambda may depend on Q
        if (lambda > Lambda_max(Q) ){ // if lambda too large, increase Q
            Q = Q+deltaQ;            // Increase the Quantizer Q size
            lambda = f(Q);       // Set new Lagrange multiplier lambda
        }
    }
    if (encoder_buffer < Tempty){        // Buffer threatens to underflow?
        lambda=lambda-deltalambda;       // Yes, so decrease lambda
        if (lambda < Lambda_min(Q) ){    // if lambda too small, decrease Q
            Q = Q-deltaQ;                // Decrease Quantizer Q size
            lambda = f(Q);           // Set new Lagrange multiplier lambda
        }
    }
    if (not last pixelblock) then goto start_encoding_pixelblock;    //Next block
```

// Done with picture.

Variations on this general rate control algorithm could include multiple different thresholds for the encoder_buffer value, whereby if encoder_buffer greatly exceeded the Tfull threshold then Quantizer Q could be incremented immediately without waiting for the Lagrange multiplier lambda to exceed its threshold. Similarly, if encoder_buffer was significantly below the Tempty threshold then the Quantizer Q could be decremented immediately. Alternatively, the deltalambda step size could be increased if encoder_buffer greatly exceeded the Tfull threshold or greatly undershot the Tempty threshold.

The values of deltalambda and deltaQ might vary with the quantizer Q or with video picture type (I-picture, P-picture, or B-picture). Furthermore, the increment operation on the Lagrange multiplier lambda might be replaced by a multiplication that could change the Lagrange multiplier lambda by a certain percentage amount. For example, the Lagrange multiplier lambda may be changed using the following equation for a lambda increase operation:

$$\text{lambda} = (1 + \text{deltalambda}) \times \text{lambda}$$

Similarly, for the lambda decrement operation the following equation may be used

$$\text{lambda} = (1 - \text{deltalambda}) \times \text{lambda}$$

This simple rate control algorithm illustrates the use of varying lambda for this application. Other more complicated rate control algorithms have also been devised, and those other rate control algorithms too could benefit from varying the Lagrange multiplier lambda.

Visual Distortion Tradeoff

Another application for varying the Lagrange multiplier lambda is in the use of visual distortion criteria. The distortion D is often measured by summing the squared difference between the original pixel values and the decoded pixel values. However, this simple distortion measurement method is not well tuned to the actual visibility of pixel errors in a video picture. Thus, such a simple distortion measurement method may cause the preceding minimizations to provide give less than optimal results. Thus, an algorithm that takes subjective effects into account is often more useful.

The visibility of encoding noise may be taken into account by calculating a visual mask value M for each pixelblock or subblock that will be encoded in the video picture. The visual mask value M is based on spatial variations and temporal variations of the pixels within the region.

A larger value of visual mask M indicates greater masking that makes the distortion more difficult to visually detect. In such regions, the distortion D can be increased and the bit rate R reduced. This is conveniently accomplished by using M × lambda (the Lagrange multiplier) in the encoding optimization algorithm instead of the Lagrange multiplier lambda alone. The following pseudo code sets for the modified algorithm:

```
Start_encoding_picture:          // Begin encoding a video picture
        input desired D;         // Get the desired Distortion D value
        find D_Q nearest to D;   // Find the D value closes to the desired D
        Qnorm = h(D_Q);          // Determine normal Q with no masking
        lambda = f(Qnorm);       // Determine the Lagrange multiplier lambda
start_encoding_pixelblock :      // Begin encoding a pixelblock from the picture
        Q = Qnorm;               // Set Q to the normal Q with no masking
        calculate visual mask M; // Determine the visual masking amount
        while(M×lambda > Lambda_max(Q) ){   // if strong masking, increase Q
                Q = Q+deltaQ;               // Raise the Quantizer Q size
        }
        code pixelblock( M×lambda, Q );  // Encode using M×lambda and Q
        if (encoder_buffer > Tfull){     // If buffer threatens to fill overflow
                lambda = lambda+deltalambda;          // Increase lambda
                if ( lambda > Lambda_max(Qnorm) ){    // Test lambda
                        Qnorm=Qnorm+deltaQ; //Increase Q size if lambda too big
                        lambda = f(Qnorm);            // Calculate new lambda
                }
        }
        if (encoder_buffer < Tempty){    // If buffer threatens to fill underflow
                lambda = lambda-deltalambda;          // Decrease lambda
                if ( lambda < Lambda_min(Qnorm) ){    // Test lambda
                        Qnorm = Qnorm-deltaQ;  // Decrease Q if lambda too small
                        lambda = f(Qnorm);            // Calculate new lambda
                }
        }
        if ( not last pixelblock) then goto start_encoding_pixelblock     //Next
// Done with picture.
```

This second simple visual masking algorithm illustrates the use of varying lambda for this application. Other more complicated visual masking algorithms have also been devised, and those visual masking algorithms could also benefit from varying lambda.

Variation of the Lagrange multiplier lambda may also be useful in other encoding decisions. For example, the determination of how many B-pictures to encode when encoding a series of video pictures is often very difficult to answer. For a particular value of quantizer Q and $lambda_Q = f(Q)$, the result of encoding with one B-picture per P-

picture might be R_1,D_1 whereas the result of encoding with two B-pictures per P picture might be R_2,D_2.

If $R_2 < R_1$ and $D_2 < D_1$ then it is clear that the best answer is that two B-pictures are better. However, very often the result is $R_2 < R_1$ and $D_2 > D_1$ such that it is not clear number of B-pictures is better. In this case we can recode using two-B pictures per P picture with a smaller lambda that gives D_2 approximately equal to D_1. Then we simply compare the resulting values of R_2 and R_1 to see which bit rate is smaller.

Other scenarios may be similarly compared, e.g., interlace vs. progressive coding, coding with various motion search ranges, encoding with or without certain encoding modes, etc.

In conclusion, we present a simple but powerful method of rate-distortion tradeoff that has many applications in video coding. The foregoing has described a system for control of rate-distortion tradeoff by encoding mode selection in a multi-media compression and encoding system. It is contemplated that changes and modifications may be made by one of ordinary skill in the art, to the materials and arrangements of elements of the present invention without departing from the scope of the invention.